ANNO 1-N. 5 L. 3.900

PC 128-PC 128S

olivetti PRODEST USER

LA PRIMA E UNICA RIVISTA INDIPENDENTE PER GLI UTENTI PC 128 E PC 128 S

Numero 5-1987

Gabbiani Video News
Il giornale ecologico

Viewindex
Crea i sommari

Scriptor
Un word-processor per il PC 128

Io scrivo... a scuola

Spedizione in abbonamento postale Gruppo III/70

BASIC

OVVERO COME COMUNICARE CON IL VOSTRO PC 128S

Cominciamo da questo numero la trattazione dei comandi Basic, elencandovi in ordine alfabetico tutte le funzioni disponibili.

Il basic, come sapete, è un linguaggio molto semplice da usare, ed è anche molto facile da imparare!

La scelta di spiegare i comandi in ordine alfabetico è motivata dal fatto che le pagine di questa rivista vogliono costituire, più che una semplice lettura, un manuale, dal quale attingere informazioni preziose. Diamo il via quindi, dopo l'articolo introduttivo comparso sulle pagine del numero 4, alla spiegazione del linguaggio che vi permetterà di colloquiare con il vostro computer.

I comandi sono spiegati seguendo questo schema:

Nome comando: è il nome del comando che viene trattato.

Tipo: dà un'indicazione di massima del campo di applicazione del comando spiegato.

Sintassi: indica l'esatto modo di scrivere il comando, in modo tale che il computer possa comprendere esattamente cosa deve eseguire.

Commento: breve spiegazione delle possibili applicazioni del comando spiegato; solitamente è completato da programmi esemplificativi.

Di ogni comando trattato, tra parentesi, è riportata l'eventuale abbreviazione disponibile.

Rimbocchiamoci quindi le maniche, e iniziamo con la spiegazione dei comandi Basic!!

ABS

TIPO: Funzione di conversione.

SINTASSI: ABS (espressione).

COMMENTO:

La funzione permette di restituire il valore positivo di qualsiasi espressione numerica. L'espressione in esame può avere valori negativi o positivi.

Provate sul vostro PC 128S questi brevi programmi che vi aiuteranno a comprendere l'uso di questa funzione.

```
10 Numero = – 10
20 PRINT ABS (Numero)
```

sul video comparirà il numero 10, positivo.

```
10 Numero = 10
20 PRINT ABS (Numero)
```

il video mostrerà il valore di Numero, senza che la funzione ABS abbia provocato nulla sul valore stesso, essendo esso già positivo.

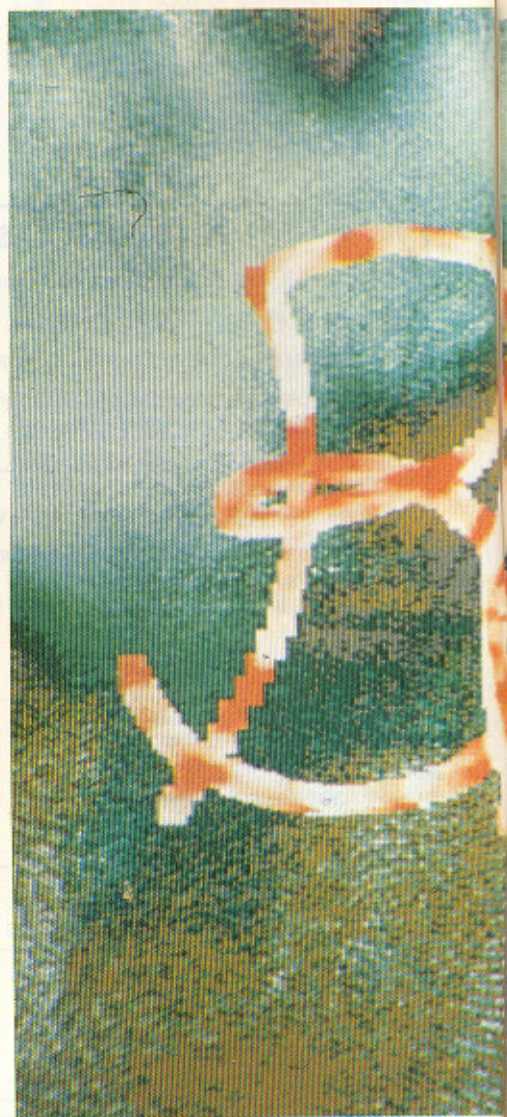
ACS

TIPO: Funzione matematica.

SINTASSI: ACS (espressione).

COMMENTO:

Questa opzione servirà soprat-

tutto a coloro i quali si occupano di matematica.

Dato un numero, compreso tra –1 e 1 questa funzione restituisce l'arcocoseno, espresso in radianti.

In questa sede non approfondiremo cosa è l'arcocoseno, in quanto è una nozione matematica che sarebbe lungo spiegare in modo esauriente.

Il breve programma riportato di seguito può aiutare a capire l'uso della funzione a chi già ne conosce le implicazioni matematiche.

```
10 angolo = 1
20 PRINT ACS (angolo)
```

ADVAL (AD.)

TIPO: Funzione di Input/Output.

SINTASSI: ADVAL (numero).

COMMENTO:

Usato per leggere o controllare dati da porte di Input/Output, quali la tastiera, la porta RS423 e altre.

L'uso del comando è di grande utilità solo se le basi di programmazione che l'utente possiede sono già solide, quindi è consigliabile non cominciare proprio da questo comando lo studio del Basic del PC 128S.

Il comando ha tre utilizzi principali spiegati di seguito uno alla volta, secondo il valore che possiede (numero).

1. (numero) = negativo. In questo caso il numero negativo corrisponde a un numero di buffer; deve essere compreso tra 1 e 9.

(numero)	porta	Input/Output	Caratteristica	
– 1	Tastiera	(lettura)	caratteri usati	0- 31
– 2	RS 423	(lettura)	caratteri usati	0-255
– 3	RS 432	(uscita)	caratteri liberi	0-191
– 4	Stampante	(uscita)	caratteri liberi	0- 63
– 5	Voce 0	(uscita)	numero byte liberi	0- 15
– 6	Voce 1	(uscita)	numero byte liberi	0- 15 (*)
– 7	Voce 2	(uscita)	numero byte liberi	0- 15 (*)
– 8	Voce 3	(uscita)	numero byte liberi	0- 15 (*)

Il simbolo (*) significa che ogniqualvolta si vuole scrivere o leggere in questo buffer sono necessari tre byte per formare un dato completo.

2. (numero) = zero.
   In questo caso il comando prende informazioni sullo stato del tasto fire del joystick.

3. (numero) = positivo.
   (numero) in questo caso rappresenta il numero del canale analogico; i valori possibili sono da 1 a 4.
   Il valore nel campo 0 –65520 deve essere incrementato a passi di 16. Al valore letto dal canale, si associa una tensione elettrica, al numero 0 corrisponde una tensione di 0 Volt, mentre il valore di 65520 corrisponde ad una tensione di uscita di 1,8 Volt, approssimativamente.

Questo comando, per poterlo usare nel modo migliore, richiede una conoscenza di base, sia del PC 128S, sia delle tecniche di programmazione proprie del linguaggio macchina.

In particolare, sull'argomento Input/Output, sarebbe possibile scrivere un libro, senza, peraltro, esaminare tutte le problematiche ad esso legate.

Importante comunque è ricordare che questo comando è messo a disposizione dal PC 128S

per controllare strumentazioni esterne, sia ad uso didattico che professionale.

AND (A.)

TIPO: Operatore aritmetico-logico.

SINTASSI: (espressione) AND (espressione).

COMMENTO:

Il principale uso di questo comando è quello di unire logicamente due eventi; un esempio in questo caso è indispensabile per capire a fondo il comando:

IF ruote = 4 AND motore = 1600 THEN PRINT "auto cc. 1600"

traducendo letteralmente:

se le ruote sono 4 e il motore è 1600 allora scrivi sul video "auto cc. 1600".

L'operatore AND è usato quindi per controllare la contemporaneità di due situazioni, quali possono essere il numero di ruote e la cilindrata di un motore. Questi due eventi permettono di 'capire' che l'automobile ha una cilindrata di 1600 centimetri cubici.

Come potete comprendere, questa funzione ha molti usi, anche più importanti di questo esempio; l'esperienza vi insegnerà più di qualsiasi ulteriore spiegazione.

ASC

TIPO: Funzione di conversione.

SINTASSI: ASC (stringa).

COMMENTO:

Questa funzione, data una stringa, restituisce il numero abbinato al primo carattere incontrato nella stringa stessa.

Questo abbinamento è fisso, ed è in pratica la tabella ASCII, tabella che potete consultare a pagina 184-185 della 'guida all'uso del sistema PC 128S', il manuale in dotazione con il vostro computer. Ricordiamo che la stringa può contenere sia lettere che numeri, e per applicazioni particolari anche caratteri diversi.

L'esempio di seguito illustrato chiarisce ogni dubbio al riguardo di questa utile funzione.

```
10 saluto$ = "ciao"
20 PRINT ASC (saluto$)
```

il programma mostrerà il numero corrispondente alla lettera c, cioè 67.

Se il numero ritornato dalla funzione è −1, allora la stringa è completamente vuota.

Comunque il numero ritornato da questa funzione rimane sempre compreso nel campo 0 - 255.

ASN

TIPO: Funzione matematica.

SINTASSI: ASN (espressione).

COMMENTO:

Ricordando il discorso fatto a proposito di funzioni matematiche complesse, riguardo la conoscenza d'uso dal punto di vista matematico, questa funzione permette di ricavare l'arcoseno.

Il campo di valori che (espressione) può accettare, senza segnalare errori, è tra i valori −1 e 1.

Il risultato verrà espresso in radianti.

Il programmino esplicativo è semplice, nonostante la difficoltà intrinseca della funzione matematica.

```
10 angolo = .89
20 PRINT ASN (angolo)
```

ATN

TIPO: funzione matematica.

SINTASSI: ATN (espressione).

COMMENTO:

Questa funzione matematica ritorna il valore dell'arcotangente, in radianti, dell'(espressione).

L'(espressione) può avere qualsiasi valore numerico. Il classico programmino d'esempio potrà aiutare chi conosce la funzione nel suo significato matematico, ma sarà di scarso ausilio a coloro i quali non possiedono ancora queste basi matematiche.

```
10 angolo = 1.45
20 PRINT ATN (angolo)
```

AUTO (AU.)

TIPO: Ausilio alla programmazione.

SINTASSI: AUTO (numero prima riga), (incremento).

COMMENTO:

Questo utile comando permette di demandare al computer l'oneroso compito della numerazione delle linee durante la fase di programmazione in Basic. (numero prima riga) deve essere un numero compreso tra 0 e 32767, inoltre dev'essere rigorosamente intero.

(incremento) è l'incremento che il comando darà alle linee.

Ad esempio: se la linea d'inizio ha il valore 1000, e l'incremento è 10, le linee generate saranno del tipo:

1010
1020
1030

e così di seguito.

Per fermare la numerazione automatica è necessario premere il tasto Escape.

Il comando Auto si fermerà automaticamente se il numero di linea supererà il valore 32767.

BGET# (B.#)

TIPO: Operatori su file.

SINTASSI: BGET# (canale)

COMMENTO:

Prende il prossimo carattere (byte) contenuto nel file: quest'ultimo doveva essere già stato aperto in precedenza.

Il comando che apre i file si chiama OPENIN, ed è bene conoscere anche questo comando per poter operare con tutta tranquillità.

(canale) è il numero dato al file attualmente aperto, dal quale intendiamo leggere i caratteri.

Il valore restituito è un numero intero compreso tra 1 e 255.

Importante è ricordare che per controllare se tutti i caratteri sono stati letti, è possibile verificare se la funzione EOF# è vera; se cioè il file è finito.

Successivamente, se abbiamo già letto l'ultimo carattere, una lettura tornerà il valore 254.

Ogni tentativo di leggere altri caratteri darà come unica conseguenza la segnalazione di un er-

rore di «fine del file».

Per un programma esemplificativo rimandiamo per completezza alla trattazione del comando OPENIN.

BPUT# (BP.#)

TIPO: Operatori su file.

SINTASSI: BPUT# (canale), (espressione).

COMMENTO:

Il comando permette di scrivere sul file definito in uscita.

(canale) è il numero associato al file nel quale vogliamo scrivere, mentre (espressione) è un valore intero compreso tra 0 e 255, ed è il codice ASCII del carattere inviato al file (vedi la funzione ASC). Il programma d'esempio sarà riportato, come già spiegato nel comando BGET#, quando verrà affrontata la spiegazione dei comandi riguardanti l'apertura dei file; tutto questo per dare una visione completa, e non limitata dell'uso dei comandi basic disponibili sul vostro PC 128S.

CALL (CA.)

TIPO: Comando.

SINTASSI: CALL (indirizzo), (variabile1), (variabile n).

COMMENTO:

Il comando mette in esecuzione una routine in linguaggio macchina, passando ad essa eventuali valori contenuti nelle variabili.

Il comando richiede una preparazione di base sul linguaggio macchina, per poter capire tutta la sua utilità e tutta la sua potenza.

(VARIABILE N) vuole significare che possono esservi più d'una variabile: quante si rendono necessarie nella logica del programma.

(indirizzo) è un numero compreso da 0 a 65535, in esadecimale 0 - FFFF.

In quel punto preciso la routine inizia, ed è quindi importante chiamare l'indirizzo esatto; una chiamata ad un indirizzo non corretto potrebbe avere effetti disastrosi per il vostro programma.

Vi è una parte di memoria chiamata 'memoria di parametri' dove sono memorizzate le informazioni relative alle variabili da passare al programma in linguaggio macchina. L'indirizzo di questa tabella è, in esadecimale, &600; mentre il formato è il seguente:

&600 Numero di variabili
&601 indirizzo della variabile numero 1
&603 tipo della variabile numero 1
&605 indirizzo della variabile numero 2
&607 tipt della variabile numero 2
&609 indirizzo della variabile numero 3
&60A è così via di seguito...

Il massimo numero di parametri è 85. il minimo zero, è quindi possibile non passare alcun parametro alla routine in linguaggio macchina, se la logica della routine non lo prevede.

I tipi delle variabili sono qui elencati di seguito, verrà dato, inoltre, ampio chiarimento su ogni punto trattato.

	Tipo	Descrizione	Esempio
1.	&00	numero di byte, intero	?a%
2.	&04	numero intero di quattro byte	a%
3.	&05	numero reale di cinque byte	a
4.	&80	una stringa ad un indirizzo	$a%
5.	&81	una variabile stringa	a$

1. Il valore che può assumere questo tipo di variabile è compreso tra 0 e 255 o tra –128 e +127 se il dato è interpretato come un numero dotato di segno algebrico.

2. In questo caso l'indirizzo punta ad un intero, formato da quattro byte, dove il byte meno significativo si trova nell'indirizzo più basso. Il numero, intero, può variare nel campo – 2147483648 + 2147483647.

3. Il numero reale è rappresentato nell'usuale formato esponente-mantissa. Il numero di byte usato è cinque. Il primo di essi è l'esponente in eccesso, può quindi variare tra – 128 e + 127; i restanti quattro formano la mantissa. Per i programmatori più esigenti è bene far notare che il segno della mantissa è dato dal bit più significativo del secondo byte.

4. L'indirizzo di una stringa è composto da un byte, varia quindi nel campo 0 - 255. Qualsiasi carattere può far parte di una stringa, escludendo il carattere ASCII 13, il return, in quanto usato dal computer per riconoscere la fine della stringa stessa.

5. L'indirizzo punta alle informazioni della stringa. In particolare l'indirizzo è composto da quattro byte i quali vengono usati per fornire tre informazioni. I primi due byte contengono l'indirizzo della memoria dove è contenuto l'inizio della stringa in esame. Il terzo byte contiene il numero di caratteri che la stringa contiene; è evidente quindi che la lunghezza massima di una stringa sarà di 255 caratteri. Il quarto e ultimo byte fornisce il numero di caratteri che la stringa contiene attualmente.

Al comando CALL possono essere passate anche variabili di tipo matriciale. Una matrice è sempre memorizzata in modo che gli elementi costituenti la matrice stessa siano conseguenti, cioè l'elemento numero 4 sarà sempre dopo l'elemento 3 e sempre prima dell'elemento 5. Questa precisazione viene fatta perché in particolari condizioni, questo tipo di memorizzazione degli elementi di una matrice non viene usato. Ricordate comunque che l'uso della funzione CALL richiede, come già detto in precedenza, approfondita conoscenza del linguaggio macchina del processore 65C12.

CHAIN (CH.)

TIPO: Comando.

SINTASSI: CHAIN (espressione).

COMMENTO:

Il comando CHAIN permette di concatenare ed eseguire da programma un altro programma.

(espressione) può essere una stringa o direttamente il nome del programma che si intende richiamare.

```
CHAIN (test)
```

Carica dalla directory corrente il programma di nome test. Anche questo modo di lavorare, illustrato di seguito, è sintatticamente corretto.

```
a$ = "test1"
CHAIN(a$)
```

CHR$

TIPO: Funzione di conversione.

SINTASSI: CHR$ (valore).

COMMENTO:

Questa funzione ritorna il carattere corrispondente al (valore).

(valore) deve essere compreso tra 0 - 255, ma i caratteri visibili sul video partono dal valore 32, il quale corrisponde allo spazio.

Ad esempio il numero 65 corrisponde alla lettera 'a', il numero 66 alla lettera 'b', e così di seguito.

Per capire meglio provate ora questo breve programma.

```
10 FOR I = 32 TO 255
20 PRINT CHR$(I)
30 NEXT I
```

sul video appariranno di seguito tutti i caratteri disponibili nel set ASCII.

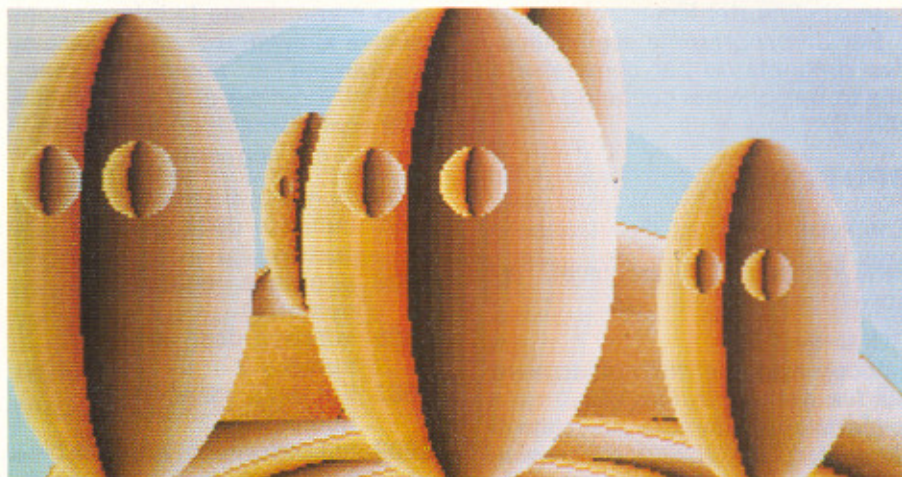
CLEAR (CL.)

TIPO: Comando.

SINTASSI: CLEAR

COMMENTO:

Il comando CLEAR cancella tutti i valori delle variabili contenuti in memoria al momento del comando.

L'unica eccezione: il tipo di variabile chiamato 'residente' non verrà interessato da questo comando. Una volta che il comando è stato dato, non esiste in basic un comando che svolga la funzione di ripristinare il valore originario delle variabili; attenzione quindi a non cancellare involontariamente tutti i vostri dati numerici dalla memoria!

CLG

TIPO: Comando.

SINTASSI: CLG.

COMMENTO:

Il comando cancella l'attuale finestra grafica mostrando il video interamente riempito del colore di fondo scelto.

Questo comando è l'unico che permette di cancellare un disegno. Per cancellare il video selezionato nel modo scrittura caratteri, il comando da usare è di CLS, riportato e spiegato più avanti.

CLOSE# (CLO.#)

TIPO: Operatori su file.

SINTASSI: CLOSE# (canale).

COMMENTO:

Il comando CLOSE# permette di chiudere un file aperto. La scelta di quale file chiudere è selezionabile dando il corretto valore a (canale). (canale) è un numero che viene associato ad un particolare file in fase di apertura.

Esempio dell'uso di questo comando è fatto nella trattazione del comando OPENIN, questo per dare una visione completa dell'uso dei file nei vostri programmi.

CLS

TIPO: Comando.

SINTASSI: CLS

COMMENTO:

Questo comando cancella tutto il video nel modo caratteri, il modo disponibile all'accesione della macchina.

È bene usarlo sempre prima di ogni programma basic. Provate i due programmi riportati di seguito che vi faranno capire l'utilità di questo semplice comando.

```
10 PRINT "comincia dalla
          posizione attuale"
20 PRINT "del cursore, ma dove
          è il"
30 PRINT "cursore in questo
          momento?"
```

la scritta comincia da dove il cursore era già posizionato; le parole scritte in precedenza sul video vengono solo spostate verso l'alto, ma non cancellate.

```
10 CLS
20 PRINT "comincia dalla
          posizione attuale"
30 PRINT "del cursore, ma dove
          è il"
40 PRINT "cursore in questo
          momento?"
```

in questo caso la scritta verrà visualizzata sul video a cominciare dall'alto, e i caratteri presenti prima, come potete vedere se provate il programma sul computer, sono stati cancellati.

COLOR o COLOUR (C.)

TIPO: Comando.

SINTASSI: COLOR (espressione)

Colore	Modo 2	Modo 4	Modo 16
0	nero	nero	nero
1	bianco	rosso	rosso
2	nero	giallo	verde
3	bianco	bianco	giallo
4	nero	nero	blu
5	bianco	rosso	magenta
6	nero	giallo	celeste
7	bianco	bianco	bianco
8	nero	nero	nero/bianco
9	bianco	rosso	rosso/celeste
10	nero	giallo	verde/magenta
11	bianco	bianco	giallo/blu
12	nero	nero	blu/giallo
13	bianco	rosso	magenta/verde
14	nero	giallo	celeste/rosso
15	bianco	bianco	bianco/nero

COMMENTO:

Il comando spiegato ora è necessario se il programmatore desidera usare particolari colori per evidenziare messaggi sul video.

(espressione) può variare tra 1 e 255.

I numeri tra 1 e 15 settano il colore dei caratteri scritti sul video; aggiungendo il numero 128 si setterà il colore di fondo del testo.

Il numero di colori disponibile è in funzione del tipo di modo usato.

La tabella qui sotto servirà a chiarire qualsiasi dubbio; inoltre sarà un utile riempimento a tutti coloro che fanno spesso uso del colore nei programmi.

Nel modo a sedici colori, i colori dall'ottavo al quindicesimo compreso sono lampeggianti.

Ad esempio, settando il colore numero 14, il testo verrà scritto in celeste e, lampeggiando, diverrà alternativamente rosso-celeste.

Per cambiare il modo di visualizzazione dei colori fare riferimento al comando MODE.

Provate questo programma che vi scriverà lettere nere su fondo bianco:

```
10 COLOR 1
20 COLOR 128
```

COS

TIPO: Funzione matematica.

SINTASSI: COS (espressione)

COMMENTO:

Questa funzione matematica restituisce il coseno di un angolo espresso in radianti.

Il valore restituito è di tipo reale compreso tra −1 e +1.

Il programma sotto proposto mostra uno dei possibili usi di questa funzione.

```
10 angolo = 3.1415926
20 PRINT COS (angolo)
```

il numero visualizzato dovrà essere −1.

COUNT

TIPO: Ausilio alla programmazione.

SINTASSI: COUNT (COU.)

COMMENTO:

Il comando appena analizzato, permette di avere un aiuto, specialmente nella stampa di tabelle. COUNT avrà, infatti, il valore corrispondente al numero dei caratteri stampati dopo l'ultimo return, valore compreso nel campo 0 - 255.

Il valore di questa variabile di sistema viene incrementato automaticamente ogni volta che un'istruzione PRINT, INPUT o REPORT viene eseguita; non viene modificato il valore se si usa un comando VDU o qualsiasi altro comando grafico.

COUNT assume il valore zero dopo ogni istruzione tipo CLS o MODE.

È possibile usare il comando anche per scrivere un definito numero di caratteri.

Un esempio è qui di seguito riportato.

```
10 REPEAT PRINT "=";
20 UNTIL COUNT = 30
30 SCRITTI = COUNT
```

DATA (D.)

TIPO: Comando.

SINTASSI: DATA (dato1), (dato2), (dato...)

COMMENTO:

questo comando deve essere usato sempre in congiunzione con il comando READ.

(dato1), (dato2), (dato...) significa che possono essere elencati una serie di dati che potranno es-

sere letti in seguito dal comando READ.

Se vogliamo avere in memoria, e quindi sempre disponibili i mesi dell'anno, la linea DATA potrà avere la seguente forma:

10 DATA gennaio, febbraio, marzo, aprile, maggio, giugno
20 DATA luglio, agosto, settembre, ottobre, novembre, dicembre

Ricordate che per accelerare quanto più è possibile l'esecuzione dei programmi basic, le linee DATA devono trovarsi sempre all'inizio del programma.

Infatti l'interprete basic cerca nel programma i DATA a partire dall'inizio del programma presente in memoria.

Nei DATA possono convivere pacificamente anche variabili di tipo diverso, sarà compito infatti delle linee dove sono contenuti i comandi READ gestire correttamente la lettura delle diverse variabili.

DEF

TIPO: Comando.

SINTASSI: DEF FN(nome), (lista variabili)
DEF PROC(nome), (lista variabili)

COMMENTO:
DEF viene usato per definire una funzione o una procedura.

Questo comando deve essere posto in testa a tutta la procedura, deve essere quindi obbligatoriamente posto come prima riga della stessa.

(nome) è il nome che si vuole dare alla procedura, richiamabile poi in tutto il programma solo con il nome.

(lista variabili) rappresenta l'insieme dei parametri che si vogliono trasferire da e per la procedura stessa, in modo da poter usare una sola procedura generale per lo svolgimento di compiti simili e ripetitivi.

DEG

TIPO: Funzione matematica.

SINTASSI: DEG (valore)

COMMENTO:
Questa funzione matematica dà come risultato un numero reale uguale a questa espressione 180*N/PI; dove PI è il valore della costante pigreco (3.1415926) ed N è il numero introdotto come (valore).

Converte quindi la misura di un angolo effettuata in radianti in un valore espresso in gradi.

Esempio:

10 PRINT DEG (PI/2)

sul video apparirà il valore in gradi dell'angolo retto; con una certa approssimazione dipendente dalla precisione di PI (pigreco).

DELETE (DEL.)

TIPO: Ausilio alla programmazione.

SINTASSI: DELETE (numero iniziale), (numero finale)

COMMENTO:
Questo comando và usato con la certezza di conoscere con precisione la sua funzione. Questo per evitare di perdere parti di programmi in memoria.

(numero iniziale) e (numero finale) devono essere degli interi compresi nel campo 0 - 32767.

Questi numeri sono numeri di linea del programma che avete nella memoria di lavoro del basic.

(numero iniziale) è il primo numero di linea che si intende cancellare, che deve essere sempre più grande del (numero finale).

(numero finale) è il numero finale delle linee che intendiamo cancellare. Quando questo comando viene confermato, le linee di programma comprese tra i due limiti vengono irrimediabilmente cancellate. Fate molta attenzione quindi ai numeri di linea e usate questo comando con prudenza, è pericoloso!!

DIM

TIPO: Comando.

SINTASSI: DIM (variabile(spazio riservato))

COMMENTO:
Questo comando permette di dimensionare, di riservare cioè una certa quantità di memoria, alla (variabile). Deve essere specificato lo spazio necessario alla (variabile), e se lo spazio è eccessivo vi verrà segnalato un errore di limitata disponibilità di memoria.

La lunghezza di una qualsiasi stringa non deve essere dimensionata, in quanto una stringa può avere una lunghezza massima di 255 caratteri.

Solo le variabili intere o matrici, anche bidimensionali devono usare l'istruzione DIM; essa va posta sempre all'inizio del vostro programma.

```
10 DIM a (10)
20 FOR I = 1 TO 11
30 A(I) = I
40 PRINT A(I)
50 NEXT I
```

questo programma è sbagliato, infatti la matrice di nome (a) è stata dimensionata per contenere 10 valori numerici, ma abbiamo voluto assegnarne 11. La correzione del programma è possibile:

1. modificando la linea 10 in questo modo:

   10 DIM a (11)

   per contenere anche l'undicesimo valore numerico.

2. cambiando la durata del ciclo FOR-NEXT:

   20 FOR I = 1 TO 10

   per assegnare solo 10 numeri, tanti quanti la variabile può contenere.

DIV

TIPO: Funzione matematica.

SINTASSI: (espressione1) DIV (espressione2)

COMMENTO:
Il comando dà il quoziente della divisione tra (espressione1) e (espressione2) trascurando il resto dell'operazione matematica.

Il quoziente non è esatto, è un valore intero, e quindi arrotondato.

(espressione2) deve essere diversa da zero, altrimenti verrà segnalato un errore dal computer.

Esempio:

```
10 num1 = 100
20 num2 = 25
30 PRINT num1 DIV num2
```

sul video verrà visualizzato il numero 4.

Se invece di 100 la variabile num1 è 119, il numero visualizzato dal computer sarà sempre 4; infatti come già detto poco sopra, questa funzione non ritorna valori esatti, ma interi approssimati per difetto.

Per verifica provate questo programma:

```
10 num1 = 10
20 num2 = 25
30 PRINT num1 DIV num2
```

sul video dovreste leggere sempre il numero 4.

DRAW (DR.)

TIPO: Comando.

SINTASSI: DRAW (espressione1), (espressione2)

COMMENTO:

Il comando, usato per il tracciamento di linee, parte dalle coordinate in cui si trova il cursore grafico attualmente, e si porta nelle coordinate date dai valori di (espressione1) e (espressione2). I limiti dei valori di (espressione1) e (espressione2) sono −32768/+32767.

EDIT (ED.)

TIPO: Ausilio alla programmazione.

SINTASSI: EDIT (linee), (stringa)

COMMENTO:

L'uso di questo comando permette di usare l'editor per correggere testi basic.

Per capire l'importante funzione che questo comando svolge è bene sapere cos'è un Token.

Ogni comando basic presente in un programma in memoria viene convertito in un numero. Questo numero (Token appunto) permette al computer di riconoscere tra loro i comandi e, dando un codice a ciascuno di essi, consente un notevole risparmio di spazio.

Per esempio: il comando PRINT viene convertito nel valore esadecimale &F1.

L'editor prima di mostrare il testo «traduce» i Token in parole chiave basic.

A questo punto, quando il testo è pronto in memoria nel suo formato «tradotto», potete lavorare come con qualsiasi altro testo.

Quando poi terminerete il lavoro l'editor si incaricherà di tradurre nuovamente, da parola estesa (PRINT) a codice Token (&F1).

(linee) sono rispettivamente il numero iniziale e quello finale della parte del programma basic che desiderate correggere o rivedere.

(stringa) è un parametro che può essere omesso: è usato per far selezionare all'editor solo le linee di programma che contengano la (stringa), effettuando una ricerca in tutto il programma basic. Da notare che ambedue i testi saranno presenti in memoria, quindi se il programma basic è molto lungo, potrebbero nascere problemi di memoria.

A questo punto, se un errore di memoria è stato segnalato, è necessario dare il comando CLEAR.

ELSE

TIPO: Comando.

SINTASSI: ELSE (comando)

COMMENTO:

Il comando ELSE può essere usato in una parte qualsiasi del programma, ma solo in congiunzione con la funzione IF ... THEN ... ELSE.
Inoltre può essere usato anche con il comando ON. spieghiamo ora i due fondamentali usi di questo comando per mezzo di programmi d'esempio in basic.

Modo d'uso 1

```
10 a = 1
20 IF a = 1 THEN PRINT "A vale UNO" ELSE PRINT "A diverso da UNO".
```

Provate a dare il RUN a questo programma, sul video comparirà la scritta "A vale UNO", ma se cambiate il valore nella linea 10, e al posto del numero 1 mettete un qualsiasi altro valore, sul video apparirà la scritta "A diverso da UNO".

Modo d'uso 2

```
10 scelta = 1
20 ON scelta GOTO 100,200,300,400 ELSE PRINT "scelta errata"
```

se il valore di (scelta) è compreso tra 1 e 4 allora la sequenza del programma si sposterà alle linee indicate dopo il GOTO.

Nel caso che (scelta) abbia un valore fuori dal campo 1-4 allora in questo caso sul video apparirà la scritta "scelta errata".

END

TIPO: Comando.

SINTASSI: END

COMMENTO:

Questo comando indica al computer la fine di un programma basic.

Il computer si ferma comunque sempre all'ultima linea del vostro programma, ma la logica può richiedere la fine dopo che particolari condizioni sono state incontrate.

ENDPROC

TIPO: Comando.

SINTASSI: ENDPROC

COMMENTO:

Il comando definisce la fine di una procedura definita dall'utente.

Quando chiamato, esso causa l'interruzione della procedura attualmente in uso e richiama le variabili globali, cioè le variabili in comune con il programma basic principale.

Il programma continuerà, passando il controllo al comando che si trova subito dopo quello che ha chiamato la procedura stessa.

Questo comando deve essere usato per chiudere una procedura, altrimenti un errore di mancata chiusura di procedura vi verrà segnalato.

LA TASTIERA DEL PC 128S

Il sistema principe per comunicare con il vostro computer

La tastiera è stata per anni l'unico sistema disponibile per poter colloquiare con il computer. Ultimamente invece si sono affacciati sul mercato dell'informatica nuovi sistemi per comunicare con l'elaboratore: la comunicazione vocale, i monitor sensibili al tatto e, il più utilizzato attualmente, il Mouse.

Il Mouse, chiamato così perché ricorda vagamente la forma di un topo, è l'accessorio che ha riscontrato più successo come possibile sostituto della tastiera. I principali vantaggi che si possono riscontrare nell'uso del Mouse sono: la facilità d'uso e la maggiore velocità di colloquio con il computer.

Molti programmi offrono, nelle loro opzioni, l'uso del Mouse; specialmente quando si tratta di effettuare scelte all'interno di un programma.

Un esempio di programma per il PC 128S che usa il Mouse, è il programma Welcome.

Tutti i fortunati possessori del PC 128S posseggono questo programma, il quale è un tipico esempio di come l'uso del Mouse risulti facile e insostituibile. Per certi tipi di lavori, comunque, la tastiera è ancora indispensabile; pensate, infatti, a tutti i problemi legati all'elaborazione di testi.

L'uso del Mouse sarebbe impossibile, e la sostituzione della tastiera con altri accessori è, per ora, decisamente improbabile.

Nel vostro PC 128S vi è una presa sul retro dove potete inserire il Mouse, senza bisogno di particolari accorgimenti di installazione, e senza bisogno di caricare programmi dedicati alla sua gestione.

Anche il vostro PC 128S è quindi al passo con i tempi, offrendovi quanto di meglio la tecnologia metta a disposizione attualmente.

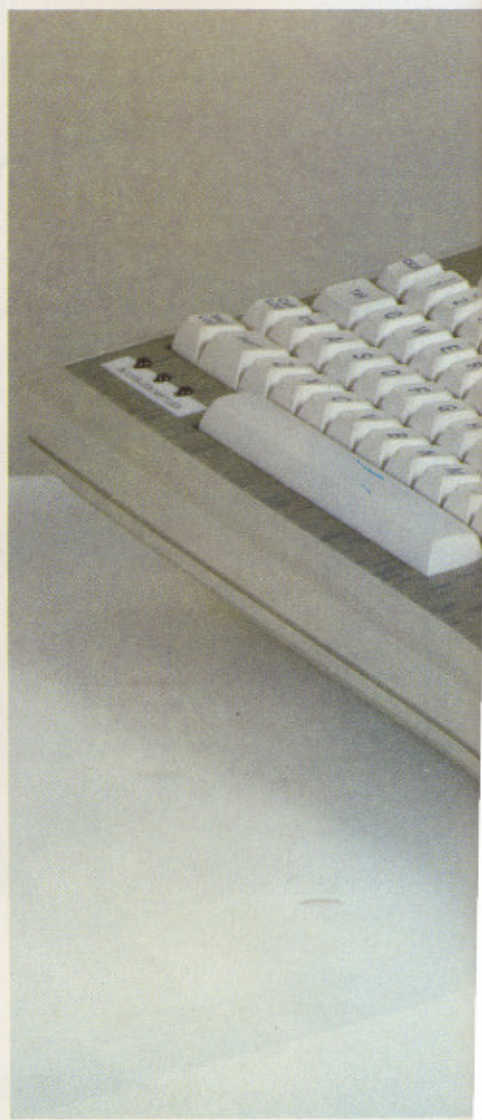
Cos'è la tastiera

La tastiera, fondamentalmente, è una serie di interruttori, come quelli solitamente usati per accendere e spegnere la luce; la differenza consiste nel fatto che il tocco è più morbido e la funzione è naturalmente diversa.

Quando voi premete il tasto corrispondente, per esempio, alla lettera A, il circuito elettronico dedicato al controllo della tastiera capisce quale tasto è stato premuto, e lo comunica al MICROPROCESSORE.

Esso si incarica poi di visualizzare il carattere sul monitor.

L'operazione di controllo, da parte del computer dell'avvenuta pressione di un tasto, è eseguita molte volte in un secondo: ciò per evitare che una veloce battitura dia luogo ad una perdita di caratteri.

In questo articolo, comunque, tratteremo la tastiera in sé, e non le circuterie elettroniche che ne

permettono il funzionamento: argomento questo che potrà essere sviluppato in un prossimo numero, visto l'interesse che molti di voi hanno di conoscere in modo approfondito il proprio strumento di lavoro.

Divisione della tastiera

La tastiera del vostro PC 128S può essere divisa, in quattro parti, descritte qui di seguito in modo approfondito; intendiamoci non si possono dividere fisicamente, ma solo logicamente!.

1. La tastiera ALFANUMERICA: è in pratica una serie di tasti simili a quelli di una macchina da scrivere con l'aggiunta, però, di alcuni altri. I tasti che solitamente non trovate sulla tastiera di una macchina da scrivere sono: il Return, il CTRL, il tasto Escape. Il termine «alfanumerico» è usato per indicare una serie di caratteri: i quali possono essere sia lettere che numeri.
2. Il TASTIERINO NUMERICO: contiene tasti per scrivere numeri, inoltre vi sono anche tasti relativi alle varie operazioni matematiche quali la somma (+), la sottrazione (–), la divisione (/) e la moltiplicazione (*).
3. I tasti CURSORE: sono un gruppo di tasti che assolvono la funzione di controllo dei movimenti del cursore. In molti programmi applicativi l'uso di questi tasti è indispensabile: basti pensare al programma View, dove per poter redarre un testo in modo semplice, il cursore deve poter essere posizionato in modo veloce in una qualsiasi parte del documento.
4. I tasti FUNZIONE: sono dieci tasti posti nella parte alta della tastiera, ai quali i vari programmi assegnano, secondo esigenze particolari, compiti diversi. Possiamo citare, come esempio sempre il View, che fa largo uso dei tasti funzionali.

Caratteristiche da ricordare

Facciamo insieme una veloce, ma chiara, carrellata sui tasti che normalmente non sono presenti sulle macchine da scrivere, cercando di spiegarne il funzionamento.

La pressione che il tasto richiede per chiudere l'interruttore, e stabilire quindi il contatto, è quella normalmente richiesta da qualsiasi tastiera di una macchina da scrivere elettronica: pertanto deve essere breve e leggera.

È importante sapere che la vostra tastiera, molti se ne saranno già accorti, permette la ripetizione automatica dell'ultimo tasto premuto, se non viene rilasciato.

Attenti quindi quando scrivete!

Nella parte sinistra della tastiera, in basso, ci sono tre lucette, comunemente chiamate «SPIE», le quali segnalano se le funzioni 'CAPS LOCK' e 'SHIFT LOCK' sono attive.

La prima spia a sinistra indica, se accesa, che il computer è collegato alla linea elettrica, ed è pronto ad essere usato.

Spieghiamo ora la funzione del tasto chiamato 'CAPS LOCK'.

In italiano si può tradurre con 'BLOCCA MAIUSCOLE', e la sua funzione è quella di far scrivere tutte le lettere in maiuscolo; la stessa funzione si ottiene premendo il tasto 'SHIFT LOCK', letteralmente 'BLOCCA ALTO'.

La differenza tra i due tasti è la seguente: il 'CAPS LOCK' serve a

scrivere solo le lettere maiuscole, mentre per scrivere i caratteri posti in alto sul tasto è necessario il tasto SHIFT. Facciamo un esempio che spieghi questo concetto, più semplice a farsi che a dirsi.

Il tasto accanto alla lettera 'L', presenta due caratteri distinti: il segno '+' e il punto e virgola. Per scrivere il punto e virgola è sufficiente premere il tasto, ma per scrivere il + è necessario premere il tasto SHIFT insieme al tasto '+;'.

Il tasto SHIFT lo si può trovare in due luoghi diversi: in basso a sinistra vicino alla lettera 'Z' e in basso a destra vicino al tasto '?/'.

Distinguiamo quindi due casi, che adesso proviamo sul computer per capire meglio quanto sopra esposto, e per non lasciare alcun dubbio nella mente di alcuno:

1. Se vogliamo la lettera L maiuscola sarà necessario premere o il tasto SHIFT + l o premere il tasto 'CAPS LOCK' e successivamente la lettera l; in quest'ultimo caso, è necessario far attenzione, perché si continuerà a scrivere in maiuscolo sino a quando non si premerà nuovamente il tasto 'CAPS LOCK'.
2. Se invece desideriamo scrivere il punto di domanda (in basso a destra), occorrerà premere prima il tasto SHIFT e contemporaneamente il tasto '?/'.

Ricordatevi che i tasti 'CAPS LOCK' e 'SHIFT LOCK' rimangono attivi fino a quando non vengono disattivati premendoli nuovamente.

Per capire se i tasti sono attivi basta guardare che le corrispondenti spie siano accese.

Per complicarvi ancora di più la vita, come non bastasse, ora vi spieghiamo un'altra caratteristica interessante.

Nel caso che il tasto 'CAPS LOCK' sia attivo, scriverete solo maiuscole, come ormai sapete, ma premendo il tasto SHIFT e contemporaneamente una lettera qualsiasi, scriverete momentaneamente in minuscolo.

Da notare che il vostro PC 128S non vi permetterà mai di tenere le funzioni 'CAPS LOCK' e 'SHIFT LOCK' attive contemporaneamente.

Passiamo ora al tasto CTRL (CONTROL), il quale non ha alcun effetto se usato da solo, ma in congiunzione con altri tasti produce 'effetti speciali'.

Provate ad esempio l'effetto che si ottiene premendo CTRL 'G'.

Avete sentito qualcosa? No? Allora non avete realmente provato sul computer, andate, provate e saprete cosa succede.

Dato che ci siete, provate anche l'effetto della combinazione dei tasti CTRL 'L'.

Vedete ancora qualcosa sul video?

Altro tasto che analizziamo è il tasto DELETE, la cui funzione è quella di cancellare l'ultimo carattere scritto sul video.

Il tasto TAB svolge la stessa funzione che è propria della barra spaziatrice (il tasto più lungo della tastiera).

Alcuni programmi, nel caso di particolari applicazioni, assegnano a questo tasto una funzione diversa. Con il tasto BREAK, invece, si provoca l'arresto del programma che attualmente è in funzione sul computer, se invece lo usiamo congiuntamente con il tasto di nome CTRL il computer verrà settato e si presenterà nella condizione in cui lo trovate quando viene acceso.

Per evitare che il tasto BREAK sia premuto accidentalmente, con le eventuali conseguenze che ne possono derivare, la Olivetti Prodest ha pensato di dotarlo di una protezione.

Vicino al tasto vi è una vite, che mediante l'uso di un cacciavite permette di disinibire la funzione.

Per ultimo, ma non ultimo, per importanza nell'uso corretto del computer, è il tasto RETURN. RETURN, che in inglese vuol dire «ritorna», è il tasto usato quando vogliamo confermare una scelta, o nel caso del sistema operativo, quando vogliamo dare un comando al computer.

Il tasto delete è usato moltissimo quando si usa il computer per programmare.

Il tasto in questione permette di eliminare l'ultimo carattere premuto, in questo caso eventuali er-

rori di battitura possono essere subito corretti.

Il comando copy, invece è più interessante, in quanto offre una possibilità che non abbiamo riscontrato né in altri computer della sua fascia di prezzo, né in computer dal costo molto superiore.

Nell'uso comune, questo comando può rivelarsi praticamente indispensabile: ma vediamo in dettaglio come si usa. Il cursore (il rettangolo luminoso che lampeggia sul video) può essere sdoppiato.

Lo sdoppiamento si verifica appunto quando si preme il tasto copy.

Con il secondo cursore, che ha la forma di un trattino, è possibile posizionarsi sulle linee dalle quali vogliamo ricopiare caratteri a noi utili.

Posizionandoci sui caratteri, che ci interessa copiare, per portarli sulla linea originaria, è sufficiente continuare a premere il tasto copy.

A questo punto abbiamo fatto una breve, ma esauriente carrellata di spiegazioni.

Per conoscere meglio la tastiera del PC 128S adesso vi diamo alcuni cenni sui set di caratteri utilizzabili. Vi sono due set di caratteri che hanno diffusione internazionale e sono precisamente:

a ASCII (AMERICAN STANDARD CODE for INFORMATION INTERCHANGE), cioè tradotto in italiano, Codice Standard Americano per lo Scambio di Informazioni.

b) TELETEXT (caratteristico dell'Olivetti Prodest PC 128S).

Questi due set sono presenti nel PC 128S, ma non possono essere usati contemporaneamente. Sono uguali sia per la parte che riguarda le lettere, che i numeri, che i simboli speciali. I caratteri corrispondono anche nel caso dei segni di interpunzione quali punto, virgola, punto e virgola doppio punto, eccetera.

Le differenze tra i due set di caratteri sono mostrate nella figura.

Possibilità offerte

È possibile crearsi un set di caratteri personale, ad esempio per uso scientifico: simboli quali il PI-GRECO ed altri; oppure definire disegni da usare nei vostri programmi, giochi o altro.

La possibilità di creare un set di caratteri personale sarà sicuramente soggetto di approfondimento in uno dei prossimi numeri di questa rivista.

Conclusioni

Speriamo di essere riusciti a farvi conoscere un po' meglio la tastiera del PC 128S, Olivetti Prodest, le sue caratteristiche particolari e utili per poter effettuare un lavoro veloce e preciso.

Se avete bisogno di altre informazioni, scrivete alla nostra redazione, saremo lieti di ritornare sull'argomento per potervi aiutare.

ADFS IL SISTEMA OPERATIVO DEL PC 128S

Terza e ultima puntata di questa breve, ma esauriente carrellata dei comandi disponibili con il sistema operativo del vostro PC 128S. Nell'articolo, abbiamo riportato esempi che possono aiutare alla comprensione di comandi che risultino particolarmente difficile da capire.

*MOUNT

ABBREVIAZIONE: *MOU.

SINTASSI: *MOUNT (drive).
COMMENTO:
Il presente comando è usato per inizializzare un disco. Inizializzare significa, in questo caso, considerare il dischetto inserito nel drive specificato in argomento; considerare naturalmente i nomi dei programmi e i file dati.

Quando si cambia il dischetto nel drive bisogna segnalare sempre tramite questo comando che è stata effettuata l'operazione.

Infatti se voi eseguite il cambio senza preavvisare il sistema operativo tramite il comando di cui sopra, il computer vi segnalerà un errore del tipo «Changed diskette».

Il presente comando è equivalente al comando *DIR:0.

*OPT1

ABBREVIAZIONE: O.1.

SINTASSI: *OPT1.

COMMENTO:
Questo comando permette di selezionare il livello di colloquio con il computer stesso.

Il controllo del colloquio avviene nei casi di salvataggio di un file o di caricamento di un file dal supporto magnetico, dal dischetto quindi.

Se questa opzione non viene scelta, durante il caricamento o il salvataggio, non viene segnalato alcun messaggio riguardante le caratteristiche del file stesso; ad esempio: la lunghezza.

*OPT4

ABBREVIAZIONE: O.4.

SINTASSI: *OPT4 (numero).

COMMENTO:
Il comando *OPT4 (numero) controlla le opzioni di caricamento automatico di file; in particolare, selezionando questo comando, il file di nome !boot verrà letto dal dischetto e contemporaneamente mandato in esecuzione.

*PRINT

ABBREVIAZIONE: *P.

SINTASSI: PRINT (nome).

COMMENTO:
Il comando PRINT (nome) permette di mostrare sul video un testo composto solo da caratteri ASCII.

(nome) è il nome che ha il file che desideriamo leggere dal dischetto.

Ricordiamo di riportare per intero il nome del file, in caso contrario il computer vi segnalerà un errore di file non trovato.

Quando troviamo un file creato da un editor, come ad esempio il View, allora sul video si leggerà il testo in modo corretto.

Se invece il file che tentiamo di leggere, non è strettamente un testo, sul video compariranno strani caratteri; sarà questa la spia per farvi capire che quel file non è leggibile.

Infatti se salviamo un testo in modo non ASCII, potremo notare la serie di caratteri assembly che caratterizza il corrispondente carattere ASCII.

*REMOVE

ABBREVIAZIONE: *RE.

SINTASSI: *REMOVE (file).

COMMENTO:
Il seguente comando permette di cancellare un file dal dischetto. Il comando è definitivo, una volta cancellato un file, è praticamente impossibile recuperare il file stesso.

Fate quindi molta attenzione a cosa cancellate!

Il comando *REMOVE è uguale, come funzione, al comando *DELETE, la differenza tra i due è che

*REMOVE non emette nessun messaggio all'operatore se il file da cancellare non esiste, mentre il comando *DELETE segnala la presenza sul dischetto del file.

Ricordate che se è stata selezionata l'opzione 1 nel comando ACCESS, non sarà possibile cancellare il programma.

Per informazioni su come proteggere e sproteggere i vostri programmi, nel numero precedente è stata fatta un approfondita spiegazione sull'uso dell'istruzione ACCESS.

*RENAME

ABBREVIAZIONE: *REN.

SINTASSI: *RENAME (nome1) (nome2).

COMMENTO:

Il presente comando permette di cambiare il nome attuale di un file presente sul dischetto. Il (nome1) viene cambiato in (nome2).

Esempio:

*RENAME testo12 testo13

il file che originariamente si chiamava testo12 non esiste più nella directory corrente, è presente invece un altro file, con lo stesso contenuto del testo12, ma di nome testo13.

Se poi vogliamo spostare un file da una directory all'altra e cambiare anche il suo nome, il comando *RENAME può assolvere anche a questa necessità.

*RENAME $.dir1.filea $.dir2.fileb

in questo esempio il file 'filea', contenuto nella directory di nome dir1, verrà copiato nella directory dir2 con il nome cambiato in fileb. Ricordate quanto detto poco sopra a riguardo delle limitazioni sulla manipolazione dei file bloccati con il comando ACCESS, al cui approfondimento, vi rimandiamo all'articolo pubblicato sul numero precedente.

*RUN

ABBREVIAZIONE: */ , *

SINTASSI: *RUN (nome).

COMMENTO:

Questo comando carica il programma (nome) e contemporaneamente lo esegue.

Il comando RUN manda in esecuzione anche programmi scritti in linguaggio macchina.

Questo è il sistema che potete usare, dal sistema operativo, per lanciare i programmi da utilizzare.

*SAVE

ABBREVIAZIONE: *S.

SINTASSI: *SAVE (nome)
(indirizzo iniziale)
(indirizzo finale)
(indirizzo esecuzione)
(indirizzo ricaricamento)

oppure:

*SAVE (nome)
(indirizzo iniziale)
(lunghezza)
(indirizzo esecuzione)
(indirizzo ricaricamento)

COMMENTO:

Il comando *SAVE permette di memorizzare una serie di dati posti in memoria.

Il dischetto conterrà una serie di dati ai quali sarà assegnato un nominativo, selezionabile ponendolo al posto di (nome).

Spieghiamo ora i parametri che possono esser scelti per salvare parti di memorie particolarmente utili nel contesto del programma che si è realizzato).

(indirizzo iniziale)

È il primo indirizzo nel quale il comando legge il dato per scriverlo su disco.

(indirizzo finale)

È l'ultimo indirizzo nel quale il comando legge i dati che devono essere trasferiti sul file (nome).

(lunghezza)

Si può specificare o l'indirizzo finale o la lunghezza del blocco di dati da trasferire, in questo ultimo caso l'indirizzo finale verrà calcolato in modo del tutto automatico.

(indirizzo esecuzione)

Da questo indirizzo il computer prenderà i dati per eseguire il programma in linguaggio macchina.

(indirizzo ricaricamento)

È l'indirizzo che seleziona l'opzione di ricaricamento.

Ricordate che tutti gli indirizzi di memoria utilizzati in questo comando, devono essere espressi sempre in esadecimale.

Queste operazioni riguardano programmi scritti prevalentemente in linguaggio macchina.

Se vogliamo salvare un programma basic è sufficiente seguire questa procedura:

*SAVE (nome del programma che si desidera salvare su dischetto!).

*SPOOL

ABBREVIAZIONE: *SP.

SINTASSI: *SPOOL (nome).

COMMENTO:

Il comando descritto viene usato per mandare tutto ciò che appare sul video, su un file specificato dal (nome).

Per terminare questo processo è necessario digitare il seguente comando:

*SPOOL

stesso comando senza però specificare alcun nome.

*SPOOLON

ABBREVIAZIONE: Non prevista.

SINTASSI: *SPOOLON (nome).

COMMENTO:

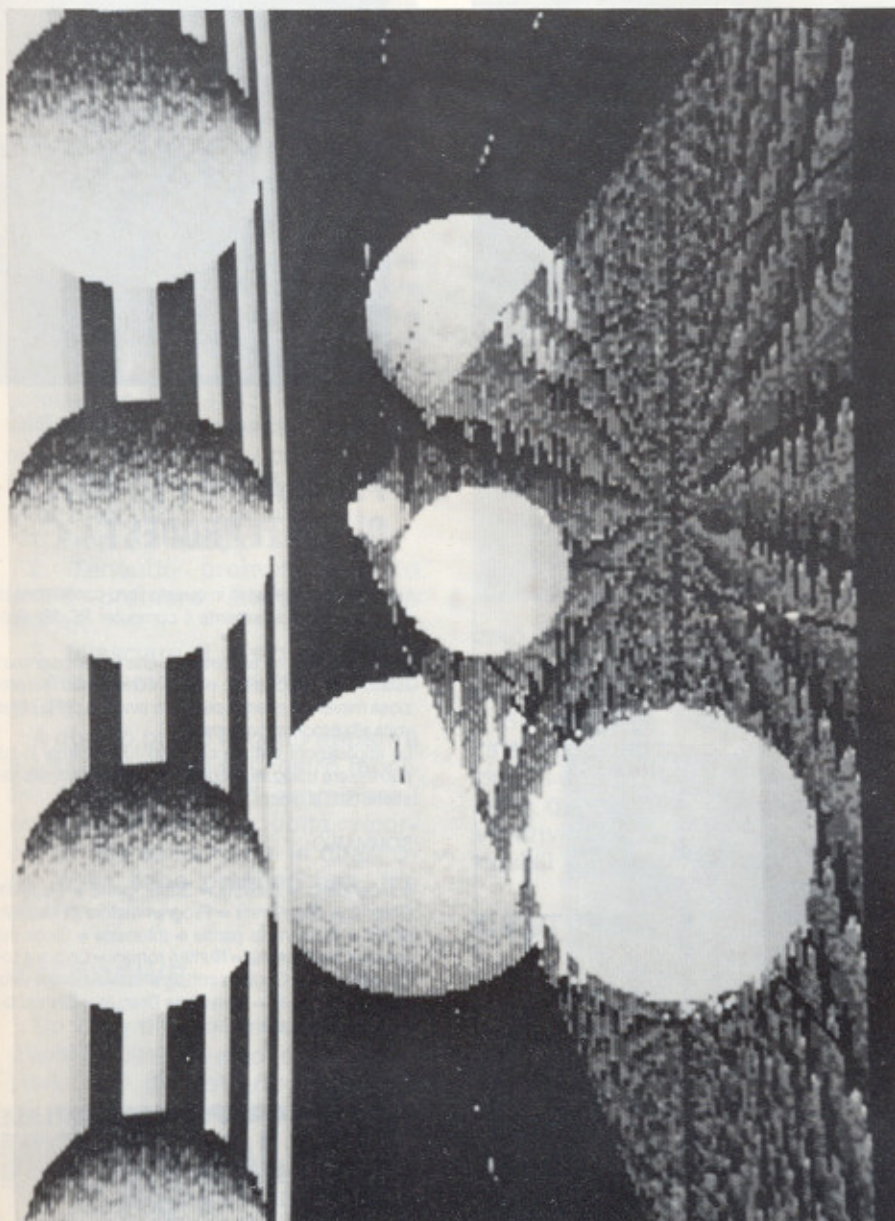
*SPOOLON permette di scrivere le informazioni che vengono inviate al monitor, in un file chiamato (nome).

A differenza del comando *SPOOL, che crea un file nuovo, *SPOOLON scrive di seguito alle informazioni già contenute nel file.

Per disabilitare la funzione del comando, è necessario dare il comando *SPOOLON senza dare alcun nome di file.

*SRLOAD

ABBREVIAZIONE: *Non prevista.

SINTASSI: *SRLOAD(nome)

COMMENTO:
Il presente comando carica in memoria, in un area ben definita, il programma o la serie di dati, denominati (nome), contenuto sul disco.

Per scegliere quello che ci interessa è necessario che (nome) sia quello del file da caricare.

Essendo il comando *LOAD molto simile rimandiamo a quest'ultimo per un migliore approfondimento delle tecniche di programmazione inerenti operazioni di Input/Output.

*SRSAVE

ABBREVIAZIONE: *Non prevista.

SINTASSI: *SRSAVE(nome).

COMMENTO:
Comando che, come il precedente, permette di eseguire operazioni di Input/Output su file presenti su dischetto. In particolare da una ben definita porzione di memoria questo comando salva dati sul dischetto dando ad essi il nome specificato in (nome).

La stessa funzione la possiamo trovare nel comando *SAVE.

*TITLE

ABBREVIAZIONE: *Non prevista.

SINTASSI: *TITLE (nome titolo).

COMMENTO:
Il comando viene usato per specificare il titolo da dare ad una precisa directory.

Come standard ogni directory ha il titolo che corrisponde al nome della direcotry stessa.

(nome titolo) è una sequenza di caratteri, i quali formano il nome da dare al titolo, devono essere in numero, minore di 19: gli altri verranno completamente ignorati.

Per confermare la stringa che deve dare il titolo alla directory, premete il tasto return.

Questo titolo verrà visualizzato ogni volta che un'istruzione *CAT verrà eseguita.

*TYPE

ABBREVIAZIONE: *TY.

SINTASSI: *TYPE(nome)

COMMENTO:
Il presente comando permette di mostrare sul video il contenuto del file specificato dal (nome).

Il file viene mostrato in un formato speciale, detto gsread. Questo modo di visualizzare i testi permette di leggere anche testi basic, omettendo però i numeri di riga, caratteristici di ogni programma.

Il formato di visualizzazione è del tutto simile a quello mostrato dal comando *LIST.

*VERIFY

ABBREVIAZIONE: *V.

SINTASSI: *VERIFY.

COMMENTO:
Il presente comando permette di verificare la formattazione dei dischetti.

Se incontrate problemi con qualche vostro dischetto, è bene utilizzare questa pratica opzione.

In particolare permette di controllare se il dischetto è leggibile in tutta la sua superficie, ed eventualmente segnala errori incontrati.

I DUE BASIC DEL PC 128

Proseguiamo con la nostra analisi del Basic relativo al PC 128

5° Parte

Eccoci qui per la quinta volta a parlare del vostro linguaggio «preferito». Allora, come proseguono i vostri viaggi nell'intricato mondo del Basic? Speriamo bene, anzi, ne siamo convinti!

Chissà quali programmi avrete redatti, grazie alle numerose scoperte fatte. Comunque è meglio non indagare troppo a fondo, rimedieremmo sicuramente una figuraccia: sarete ormai diventati talmente abili da farci impallidire!

Ma c'è ancora qualcosa che forse potremmo insegnarvi.

Eccovi quindi l'elenco dei comandi analizzati per voi in questo numero.

COPY

TIPO: Istruzione

SINTASSI: COPY*nomefile1* TO *nomefile2*

COMMENTO:

Dove *nomefile1* è il nome del file sorgente e *nomefile2* è il nome del file destinazione. Questo formato è necessario se la copia del file viene fatta sullo stesso dischetto.

Per mezzo di questo comando è possibile copiare integralmente un file da un dischetto all'altro, o sul medesimo dischetto. Ciò è possibile, sia con un drive che con due.

```
COPY"0:FILE1"TO"1:FILE2"

COPY"FILE1"TO"FILE2"

COPY"FILE1"
```

COS

TIPO: Funzione matematica

SINTASSI: COS(n.)

COMMENTO:

Dove n. può essere un numero, una funzione o una variabile che rappresenta un angolo espresso in radianti.

Il risultato ottenibile per mezzo del comando COS è il coseno di un angolo.

```
10 CLS
20 SCREEN1,7,0,1
30 LINE(0,100)–(319,100),2
40 LINE(0,50)–(319,50),4
50 LINE(0,150)–(319,150),4
60 FOR A=1 TO 319
70 ANGOLO=A/20
80 PSET(A,100–50*COS
   (ANGOLO)),0
90 NEXT A
100 END
```

CRUNCH$

TIPO: Comando

SINTASSI: CRUNCH$(stringa)

COMMENTO:

Dove stringa deve essere o una espressione stringa o una espressione numerica.

Il comando CRUNCH permette di codificare dei dati, per esempio ottenuti da un comando di input, e di renderli valutabili dal comando EVAL. Ciò permetterà di ottenere, direttamente, il risultato di un'operazione, sia che essa riguardi delle stringhe che dei numeri.

È opportuno sottolineare che il formato con cui vengono immessi i dati, deve rispettare la corretta sintassi Basic delle funzioni di cui si vuole il risultato.

Per meglio comprendere, vediamo un breve programma in cui utilizzeremo anche il comando EVAL e ciò al fine di mostrare l'interdipendenza dei due comandi.

```
10 CLS
20 SCREEN0,2,8
30 LINE INPUT"Immissione dati:
   ";A$
40 PRINT EVAL(CRUNCH$(A$))
50 ONKEY="C" GOTO10
60 LOCATE9,10
70 COLOR5,,1
80 PRINT"Per continuare pre-
   mere C"
90 PRINT CHR$(20)
100 GOTO100
```

Nel programma sopra riportato, provate ad immettere quanto segue:

```
RIGHT$("CIAO PIPPO",5)

"CIAO"+" PIPPO"

8*16+(32–3)
```

La riga 100 crea un loop, bloccando così il programma in attesa della pressione del tasto C, che consente di poter continuare ad usare l'input. Ciò determina un minimo controllo sull'introduzione dei dati, che può essere da voi e-

steso, per esempio, sul controllo dell'uscita dal programma.

La riga 90, serve ad eliminare la fastidiosa presenza del cursore lampeggiante, mentre leggiamo il messaggio del computer.

CSNG

TIPO: Funzione di conversione

SINTASSI: CSNG(n.)

COMMENTO:

Dove n. può essere sia un numero, sia una espressione numerica, sia una variabile.

Converte un numero qualsiasi in un numero a precisione singola, effettuando un arrotondamento automatico.

```
10 CLS
20 A#=(32.34565120*
45.619199)
30 PRINT CSNG(A#)
40 PRINT A#
50 END
```

CSRLIN

TIPO: Funzione

SINTASSI: CSRLIN

COMMENTO:

Ritorna il numero di linea in cui si trova il cursore mentre viene eseguita l'istruzione CRSLIN.

```
10 CLS
20 LOCATE32,13
30 PRINT CSRLIN
```

Il programma stamperà sul video il numero 13, che corrisponde al numero di riga, in posizione colonna 32 e riga 13.

CVD

TIPO: Funzione di conversione

SINTASSI: CVD(variabile stringa)

COMMENTO:

La variabile stringa deve contenere dati ottenuti tramite una conversione, per mezzo dell'istruzione MKD$(). Questa istruzione fa solitamente riferimento ad una variabile di campo precedentemente definita per mezzo dell'istruzione FIELD.

L'istruzione CVD permette la

conversione del contenuto di una variabile alfanumerica in un numero a precisione doppia.

A#=CVD(STRINGA$)

CVI

TIPO: Funzione di conversione

SINTASSI: CVI(variabile stringa)

COMMENTO:

Analoga al comando precedente, questa istruzione permette la conversione in numero intero, del contenuto della variabile alfanumerica.

A%=CVI(STRINGA$)

CVS

TIPO: Funzione di conversione

SINTASSI: CVI(variabile stringa)

COMMENTO:

Questa è la terza funzione di questo tipo, e infatti la sua sintassi e i modi sono identici a quelli dei comandi CVD e CVI: per suo mezzo si può convertire una variabile stringa in un numero a precisione singola.

A=CVS(STRINGA$)

DATA

TIPO: Istruzione di trattamento dati

SINTASSI: DATA val.1,val.2,...

COMMENTO:

Dove val.1, val.2 eccetera, sono un elenco di costanti.

Tramite l'istruzione DATA, possiamo creare un elenco di costanti, sia numeriche che alfanumeriche, di lunghezza indefinita e tramite l'istruzione READ, possiamo poi immetterle in variabili. È opportuno precisare che le variabili utilizzate dall'istruzione READ devono essere dello stesso tipo delle costanti da loro immagazzinate.

Le costanti elencate possono essere di tutti i tipi: intere, reali, a precisione singola, a precisione doppia e alfanumeriche. In tale elenco non possono comparire delle espressioni, e le costanti alfanumeriche contenenti spazi, o caratteri separatori, devono essere racchiuse fra virgolette.

Un altro comando molto utile, alla corretta gestione dei DATA, è l'istruzione RESTORE, che permette la rilettura dei DATA a partire dal numero di riga in argomento. Per verificare l'utilità del comando RESTORE provare i due programmi riportati di seguito:

```
10 CLS
20 DATA32,67,73,65,79,32,80,
   73,80,80,79
30 FOR A=1 TO 11
40 READ B
50 PRINT CHR$(B)
60 NEXT
70 GOTO 30
```

```
10 CLS
20 DATA32,67,73,65,79,32,80,
   73,80,80,79
30 FOR A=1 TO 11
40 READ B
50 PRINT CHR$(B)
60 NEXT
70 RESTORE
80 IF CSRLIN=20 THE END
90 GOTO 30
```

L'istruzione DATA è molto usata per caricare direttamente in memoria dei dati tramite l'istruzione POKE, così da ottenere delle parti di programma che lavorino direttamente in linguaggio macchina anche da Basic. Oltre a questo, la tecnica appena descritta, permette di memorizzare direttamente, in opportune zone di memoria, dei dati che in seguito potranno essere riletti con l'istruzione PEEK.

```
10 CLS
20 20 DATA32,67,73,65,79,32,
   80,73,80,80,79
30 FOR A=0 TO 10
40 READ B
50 POKE15000+A,B
60 NEXT
70 FOR A=0 TO 10
80 C(A)=PEEK(15000+A)
90 NEXT
100 FOR A=0 TO 10
110 PRINT CHR$(C(A));
120 NEXT
130 END
```

Come si può vedere, il programma sopra riportato, non è ottimizzato al meglio, e questo perché sopra tutto esiste una necessità didattica alla quale difficilmente si può far fronte diversamente.

Alla riga 50 il programma carica in memoria, a partire dall'indirizzo 15000, i valori letti dall'istruzione DATA di riga 20, tramite la riga 40. Mentre l'istruzione PEEK(), in riga 80 fa l'operazione inversa, andando a leggere i valori contenuti in memoria dalla locazione specificata in argomento.

In questo programma è altresì interessante notare l'importanza dei cicli FOR NEXT, di cui parleremo in modo più esteso in una prossima rivista, in cui chiariremo in modo inequivocabile tutti i segreti e tutte le applicazioni di tale, oserei definirle, fondamentale istruzione.

DEF FN

TIPO: Funzione

SINTASSI: DEF FNnome(var1, var2,...)= espressione

COMMENTO:

Dove nome è il nome di una variabile, o numerica o stringa, Basic e pertanto ne deve seguire le regole di formazione; var1, var2 eccetera, sono un elenco di variabili fittizie che si riferiscono all'espressione. Alla destra dell'uguale c'è l'espressione che definisce la funzione.

La funzione stessa è richiamabile tramite l'istruzione FNnome(varA, varB...), dove nome deve essere il nome di una variabile precedentemente definita con DEF, e varA, varB... eccetera, devono essere delle variabili utilizzate per passare i valori alla funzione, divenendo così il vero argomento della funzione.

Solitamente questo comando, erroneamente considerato troppo macchinoso e pertanto poco usato, permette di definire delle funzioni anche molto complesse, che possono essere poi facilmente richiamate attraverso il loro nome e l'immissione dei giusti parametri. La possibilità di far eseguire la

stessa funzione con parametri di volta in volta diversi, per esempio, in un programma di grafica che si basi su delle curve calcolabili matematicamente tramite funzioni, può far risparmiare molti passaggi e risparmiarci pertanto una notevole mole di lavoro.

Nel breve programma che segue abbiamo cercato di mostrare l'uso dell'istruzione DEF FN, sia in campo numerico che in quello alfanumerico. È importante notare l'uso delle variabili.

```
10 CLS
20 I$=" CIAO PIPPO ":
   B$="1234567890"
30 A=15: B=65: C=3: D=6:
   E=4: H=1
40 DEF FN NUM(X,Y)= SQR(X+Y)
50 DEF FN CH$(R)=MID$
   (STR$(R),2)
60 DEF FN R$(F,G)=MID$
   (I$,F,G)
70 A=FN NUM(A,B)
80 A$=FN CH$(C)
90 B$=FN R$(D,E)
100 PRINT A
110 PRINT A$
120 PRINT B$
130 FOR Q=1 TO 10
140 PRINT FN R$(Q,1)
150 NEXT Q
160 END
```

DEFDBL

TIPO: Funzione

SINTASSI: DEFDBL lettera1-lettera2,lettera3-lettera4,...

COMMENTO:

Questo comando permette di definire, in doppia precisione, un gruppo di variabili aventi la stessa lettera iniziale nel nome.

Tale istruzione, è utile porla in testa ai programmi.

DFINT

TIPO: Funzione

SINTASSI: DEFINT lettera1-lettera2,lettera3-lettera4,...

COMMENTO:

Questo comando permette di definire un gruppo di variabili, di tipo intero, aventi la stessa lettera iniziale nel nome.

L'interesse di questo comando, sta nel fatto che le variabili di tipo intero occupano molto meno spazio in memoria di quanto non ne occupino gli altri tipi di variabili, inoltre, le operazioni svolte, usando questo tipo di variabili, vengono eseguite con una rapidità decisamente superiore; e ciò non guasta mai. Pertanto, se siamo sicuri che le variabili di cui necessitiamo in un programma saranno sempre di tipo intero, ci converrà dichiararle come tali.

DEFSNG

TIPO: Funzione

SINTASSI: DEFSNG lettera1-lettera2,lettera3-lettera4,...

COMMENTO:

Questo comando permette di definire, in singola precisione, un gruppo di variabili aventi la stessa lettera iniziale nel nome.

DEFSTR

TIPO: Funzione

SINTASSI: DEFSTR lettera1-lettera2,lettera3-lettera4,...

COMMENTO:

Questo comando permette di definire un gruppo di variabili, di tipo alfanumerico, aventi la stessa lettera iniziale nel nome.

È opportuno ricordare nuovamente, che tutti i comandi di tipo DEF... vanno collocati in testa ai programmi.

DEFGR$

TIPO: Istruzione

SINTASSI: DEFGR$(n.)= n1,n2,n3...,n8

COMMENTO:

Dove n. è un numero compreso tra 0 e 127, e n1,...,n8 sono 8 numeri rappresentanti il valore di ogni singolo byte costituente il carattere ridefinito. Questi otto numeri possono essere dati sotto forma di costanti, variabili, array, eccetera, e possono essere sia decimali che binari. Quest'ultimi devono seguire le regole di rappresentazione dei numeri binari del Basic del PC 128, devono cioè essere preceduti dal prefisso &B.

Questa istruzione, deve essere obbligatoriamente preceduta dal comando CLEAR, necessario all'organizzazione della memoria utente.

Tramite DEFGR$() è possibile definire dei caratteri personalizzati o delle immagini grafiche ottenute dall'unione di più caratteri ridefiniti.

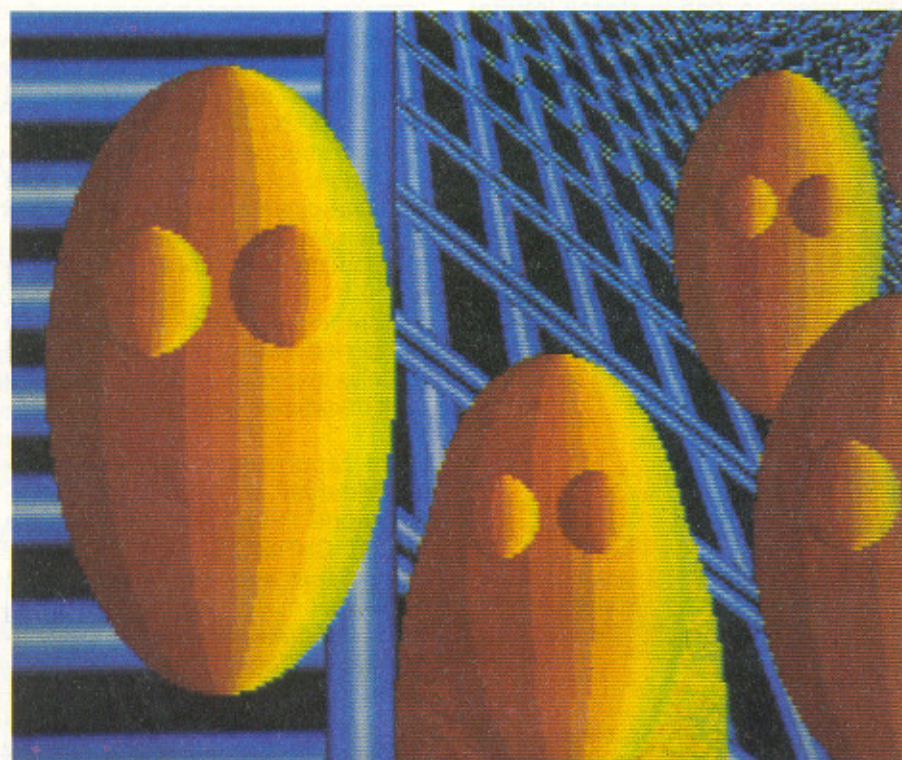
La rappresentazione dei caratteri ridefiniti, si avrà poi tramite l'istruzione PRINT GRS(n.), oppure PRINT CHR$(128+n.).

Vediamo ora, per mezzo di un breve esempio, come sia possibile ottenere delle semplici rappresentazioni grafiche, le quali però possono essere solo delle anticipazioni di creazioni molto più complesse, da voi realizzabili grazie alla potenza e flessibilità di questa tecnica.

Per prima cosa si deve disegnare un riquadro composto da 8 per 8 caselle, questa sarà la matrice da utilizzare per ridisegnare un carattere. Il perché siano necessarie 8 caselle per 8, lo dovreste oramai sapere, ma sarà opportuno chiarire nuovamente che ogni carattere visualizzato dal PC 128 è costituito da 8 byte messi in colonna uno sopra l'altro, da ciò quindi la tabella sopra descritta.

All'interno del nostro riquadro, segneremo con una crocetta ogni punto che desideriamo sia visualizzato, così da definire la nuova forma del carattere. Una volta terminata questa prima operazione, puramente creativa, alla destra di ogni riga scriveremo un numero binario in cui ripoteremo, in ordine da sinistra verso destra, con la cifra 1 i riquadri da noi segnati con una crocetta, e con lo 0 quelli vuoti. Alla fine di tale operazione ci dovremmo trovare con otto numeri binari incolonnati, rappresentanti la situazione di ogni bit dell'immagine definita.

Visto che il comando DEFGR$() accetta in argomento anche nu-

meri decimali, potremmo ora trasformare i numeri binari, appena ottenuti, nei corrispondenti numeri decimali. A tal fine, sopra ogni colonna del riquadro, dovremo segnalare, a partire da sinistra, i seguenti valori: 128, 64, 32, 16, 8, 4, 2, 1, i quali sono i valori di ogni bit, componente un byte. Questi sono determinati dalla loro posizione all'interno della serie.

Ora che siamo in possesso dei valori desiderati, possiamo finalmente scrivere un programma atto alla visualizzazione dei caratteri appena ridefiniti.

Tale programma è riportato di seguito e mostra alcune possibilità di manipolazione, dei dati ottenuti in precedenza.

```
10 CLS
20 SCREEN0,5,3
30 CLEAR,,4
40 DEFGR$(0)=24,60,126,255,
   102,102,102,255
50 DEFGR$(1)=92,42,95,124,28,
   12,12,255
60 DEFGR$(2)=&B00011000,
   &B00111100,&B01111110,
   &B11111111,&B01100110,
   &B01100110,&B01100110,
   &B11111111
70 DEFGR$(3)=&B01011100,
   &B00101010,&B01011111,
   &B01111100,&B00011100,
   &B000011,&B00001100,
   &B11111111
80 LOCATE18,10
90 PRINTGR$(0);
100 PRINTGR$(1);
110 PRINTGR$(3);
120 PRINTGR$(2);
130 A$=CHR$(128+0)+CHR$
    (128+1)+CHR$(128+2)+
    CHR$(128+3)
140 PRINT A$
150 LOCATE,5
160 FOR A=0 TO 9
170 PRINT A$
180 NEXT A
```

128	64	32	16	8	4	2	1	BIN.		DEC.
			X	X				00011000	=	24
		X	X	X	X			00111100	=	60
	X	X	X	X	X	X		01111110	=	126
X	X	X	X	X	X	X	X	11111111	=	255
	X	X			X	X		01100110	=	102
	X	X			X	X		01100110	=	102
	X	X			X	X		01100110	=	102
X	X	X	X	X	X	X	X	11111111	=	255

128	64	32	16	8	4	2	1	BIN.		DEC.
	X		X	X	X			01011100	=	92
		X		X		X		00101010	=	42
	X		X	X	X	X	X	01011111	=	95
	X	X	X	X	X			01111100	=	124
			X	X	X			00011100	=	28
				X	X			00001100	=	12
				X	X			00001100	=	12
X	X	X	X	X	X	X	X	11111111	=	255

Le righe 60 e 70 non sono altro che una ripetizione delle righe 40 e 50, esse sono state introdotte solo a pura dimostrazione, e per questo i dati riportati sono scritti in forma binaria. È interessante notare, sia l'uso del comando CLEAR, riga 30, che il trattamento dei dati, definiti dal comando DEFGR$(), identico al trattamento delle variabili stringa e ampiamente descritto dai blocchi di programma compresi tra le righe 80 e 120, 130 e 140, e per finire, 150 e 180.

Come è dato vedere, da questo semplice esempio, il comando DEFGR$() è molto potente e allo stesso tempo semplice da usare, pertanto grazie ad esso vi sarà molto facile creare animazioni, fondi, ed altro ancora.